CMR Github Workflow and FAQ

Steps:

- Fork the main repository to your GitHub account
- clone the repository from your fork
- add the main repository as a remote (e.g., git remote add upstream https://github.com/nasa/Common-Metadata-Repository.git)
- create a branch (e.g., git checkout -b feature/CMR-1234-new-thing)

- work on the branch in your fork's clone, and
- file PRs across forks to the main repository

We'll still include JIRA ticket numbers in commit messages, and continue to use JIRA as though nothing ever changed.

Useful Commands

Checking out a new branch with the latest changes from master and pushing to your fork

There are at least two ways to do this, and they are:

- fetching and resetting

> *git checkout -b ExampleBranchName*
>
> *git fetch upstream master*
>
> *git reset --hard upstream/master*
>
> *git push -u origin ExampleBranchName*

- pull latest from upstream master and check out as normal

> *git checkout master*
>
> *git pull upstream master*
>
> *git checkout -b ExampleBranchName*
>
> *git push -u origin ExampleBranchName*

Both options use the same number of commands, and accomplish the same exact thing. The only real reason to choose one over another is personal preference. If you're feeling fancy you can

BONUS: Add these functions to your .bashrc and save yourself a little bit of typing. `newbranch` takes a branch-name, and pushes a fresh branch to your fork, taken from the latest that upstream/master has.

`tomaster` simply fetches the latest that master has and resets the branch that you're currently on to that.

```
function newbranch () {
    git checkout -b $1;
    git fetch upstream master;
    git reset --hard upstream/master;
    git push -u origin $1;
}

function tomaster () {

    git fetch upstream master;

    git reset --hard upstream/master;

}
```

Once you've added that, run:

```
source ~/.bashrc
```

This will reload your bashrc and allow you to start using the function you just added.

Updating a feature branch to have the latest changes from upstream master

Again, there are a few ways to do this, but the end result can be a little different depending on what you're looking to do. The three different ways are:

- fetch upstream master and rebase your changes on top of it (rebases current branch on top of master)

  > *git fetch upstream master*
  >
  > *git rebase upstream/master*

- rebase on top of master with git pull

  > `git pull --rebase upstream master`

- merge master into your branch with git pull (git pull performs a fetch and a merge)

  > *git pull upstream master*

- interactive rebase on top of upstream/master (the preferred way to tidy up your branch before you merge, if you choose to do so)

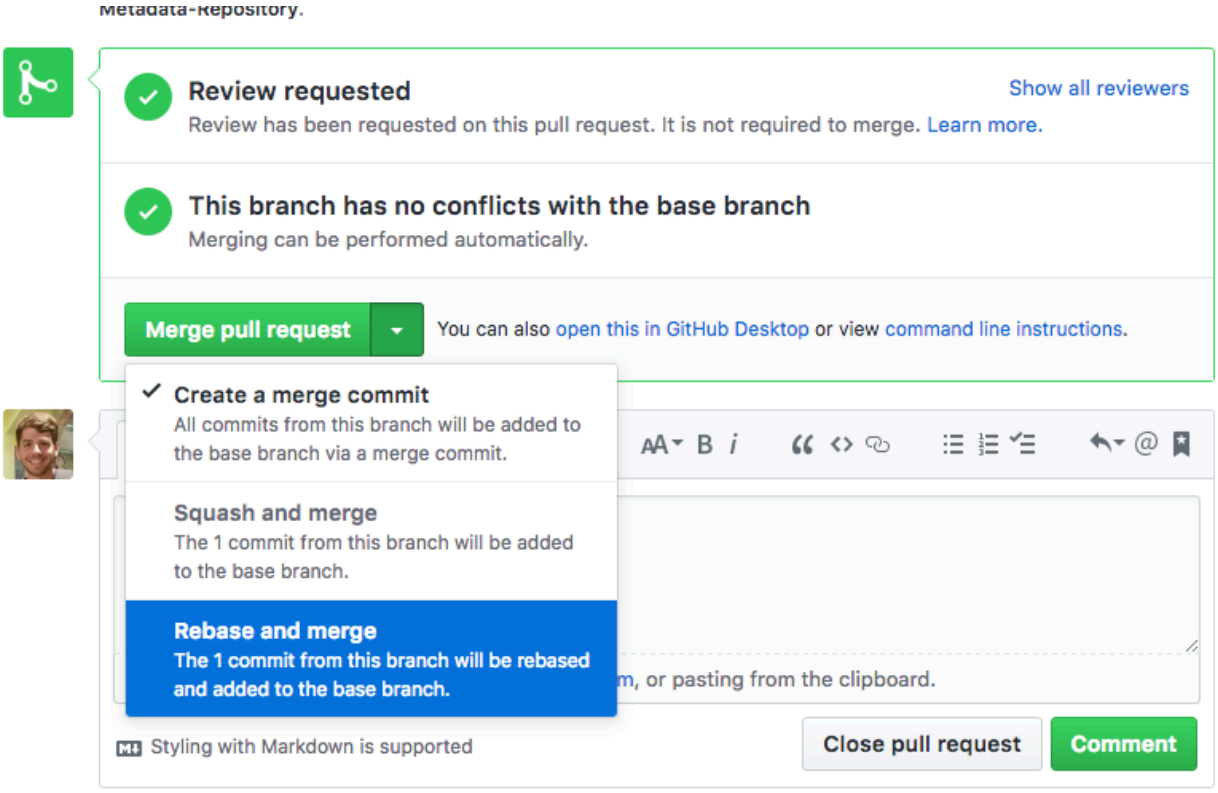
  > *git fetch upstream master*
  >
  > *git rebase -i upstream/master*
  >
  > *After you make your changes and save: git push origin <branch-name> -f*
  >
  > *Merge to master through the github interface*

Rebasing is nice because it keeps a linear history, which makes it easier to amend if needed, as well as making it easier to use git bisect to hunt down bugs. Rebasing works by replaying your commits on top of the new base. You'll also resolve conflicts the same way as you would when merging. Another nice thing that rebasing allows is to use the -i option, which allows you to squash, exclude, or mark commits as a fixup if you're like me and don't want a ton of commits in a row that just say "addressing PR comments".

Merging pull requests

The short version: use the "rebase and merge" or the "squash and merge" options. If you're feeling especially organized, interactive rebases (mentioned above) are appreciated when they make sense.

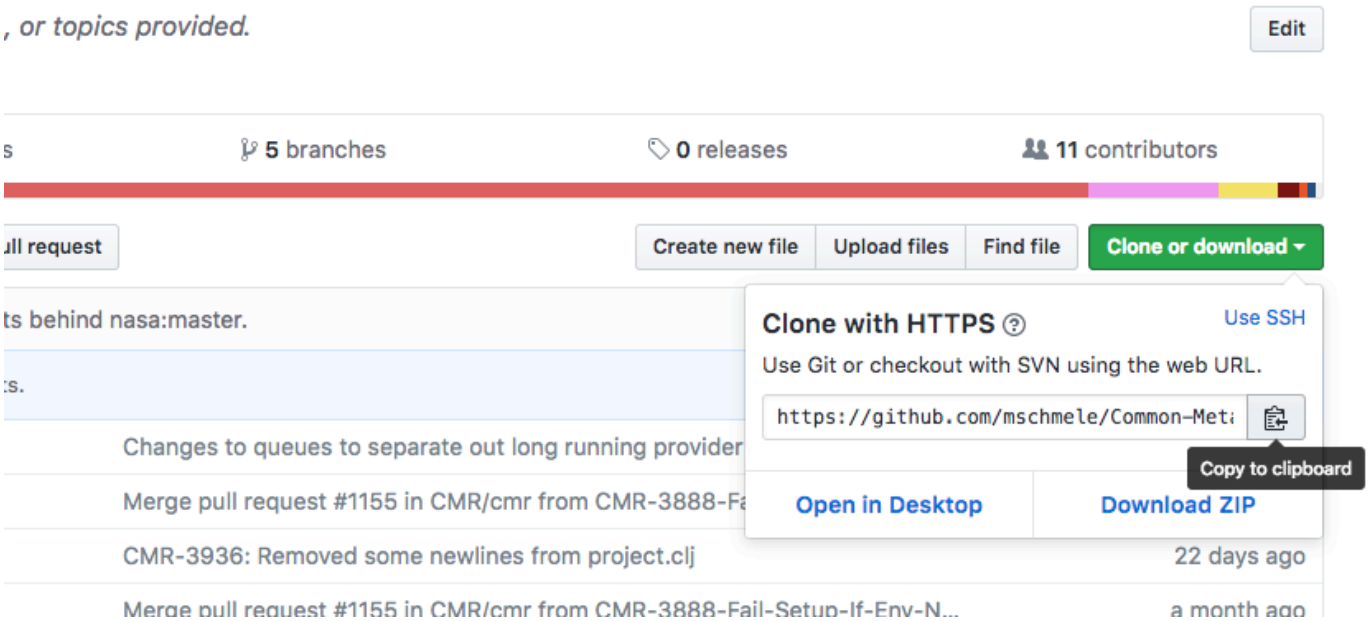
GitHub thankfully makes this easy. Click the drop-down next to the merge button, choose one of the options mentioned above, and you're off to the races. The default option (the one with the check mark next to it in the picture below) is not preferred and should be avoided.



Adding other forks as a remote to check out a team-member's branch

Navigate to the desired fork via GitHub. You can easily find a list of members at: https://github.com/nasa/Common-Metadata-Repository/network/members

Once on the proper page, find the "Clone or download" drop-down to copy the link to the remote.



In your terminal, run the following:

> *git remote add <remote-name> <link-to-remote>*

Let's say Bob has a branch that you want to check out. You've named his remote "bob", and you want to pull down his branch that is called "bobs-branch" (hypothetical Bob is very creative)

> *git fetch bob bobs-branch*
>
> *git checkout bobs-branch*

That should do the trick! You can modify code as normal, run tests, all of that. If you make changes and want to push them (using Bob's branch as an example), try the following:

Note: You'll need to have Bob give you permission to push to his fork

```
git push -u bob bobs-branch
```

This will set the remote to Bob's fork, and push the changes to it. Of course, you can always push them to your fork if you really like.

Useful articles

- Syncing a fork: https://help.github.com/articles/syncing-a-fork/
- Adding a remote: https://help.github.com/articles/adding-a-remote/
- Rebasing vs Merging https://www.atlassian.com/git/tutorials/merging-vs-rebasing